LAB # 9

RECURSIONS IN C++

Recursion is a programming technique that allows the programmer to express operations in terms of themselves. In C++, this takes the form of a function that calls itself. A useful way to think of recursive functions is to imagine them as a process being performed where one of the instructions is to "repeat the process". This makes it sound very similar to a loop because it repeats the same code, and in some ways it is similar to looping.

A function that calls itself, and doesn't perform any task after function call, is known as tail recursion. In tail recursion, we generally call the same function with return statement.

Syntax of Recursion

```
recursionfunction()
{
recursionfunction(); //calling self function
}
```

Find a Recursive Solution

There are some important steps to follow when writing a recursive function, these steps can be identified by answering the following questions:

1.What is the base case, and can it be solved?
2.What is the general case?
3.Does the recursive call make the problem smaller and does it approach the base case?

Base Case

The base case, or halting case, of a function is the problem that we know the answer to, that can be solved without any more recursive calls. The base case is what stops the recursion from continuing on forever. Every recursive function must have at least one base case (many functions have more than one). If it doesn't, your function will not work correctly most of the time, and will most likely cause your program to crash in many situations, definitely not a desired effect.

The General Case

The general case is what happens most of the time, and is where the recursive call takes place. In the case of factorial, the general case occurs when n > 1, meaning we use the equation and recursive definition n! = n*(n - 1)!.

Avoiding Circularity

Another problem to avoid when writing recursive functions is circularity. Circularity occurs when you reach a point in your recursion where the arguments to the function are the same as with a previous function call in the stack. If this happens you will never reach your base case, and the recursion will continue forever, or until your computer crashes.

C++ Recursion Example

An example to print factorial number using recursion in C++ language.

```
#include<iostream>
using namespace std;
int main()
{
int factorial(int);
int fact,value;
cout<<"Enter any number: ";
cin>>value;
fact=factorial(value);
cout<<"Factorial of a number is: "<<fact<<endl;
return 0;
}
int factorial(int n)
{
if(n<0)
return(-1); /*Wrong value*/
if(n==0)
return(1); /*Terminating condition*/
else
{
return(n*factorial(n-1));
}
}
```

Output:

Enter any number: 5
Factorial of a number is: 120

We can understand the above program of recursive method call by the figure given below:

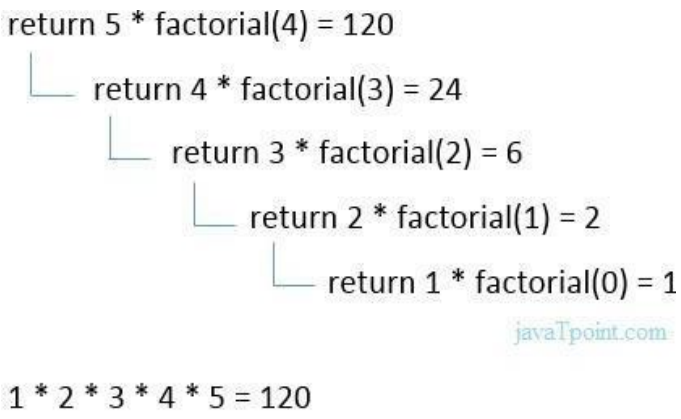

Fig: Recursion

Lab Tasks

1- Write a recursive function to obtain the first 25 numbers of a Fibonacci sequence. In a Fibonacci sequence the sum of two successive terms gives the third term. Following are the first few terms of the Fibonacci sequence: 1 1 2 3 5 8 13 21 34 55 89...

2- Develop a program that calculates the factorial of a number using recursion.

3- Develop a program that takes a word as input from the user terminated by enter key and displays the entered word in reverse order.

Input: recursion
Output: noisrucer

4- Write a program to generate and display the sum of first 10 terms of the following series using recursion:
(2 * 1) + 2, (2 * 2) + 2, (2 * 3) + 2 . . . . . .